

REGISTRO

Lílian Simão Oliveira

Registro

- Um *registro* (= *record*) é uma coleção de várias variáveis, possivelmente de tipos diferentes.
- Na linguagem C, registros são conhecidos como **structs** (abreviatura de *structures*).
- Sintaxe:
- ```
struct <nome> {
 tipo1 varival1;
 tipo2 varival2;
 tipo3 varival3;
 ...
} <nome do objeto>;
```

# Registro - Exemplo

- Um **Produto** pode ter atributos como **peso** e **preço**
- Uma **maça** é do tipo **Produto**

```
1 struct product {
2 int weight;
3 float price;
4 } ;
5
6 product apple;
7 product banana, melon;
```

# Registro – Exemplo Declaração

- Um **Produto** pode ter atributos como **peso** e **preço**
- Uma **maça** é do tipo **Produto**

```
1 struct product {
2 int weight;
3 float price;
4 } ;
5
6 product apple;
7 product banana, melon;
```

```
1 struct product {
2 int weight;
3 float price;
4 } apple, banana, melon;
```

# Registro – Exemplo Acesso

```
1 struct product {
2 int weight;
3 float price;
4 } ;
5
6 product apple;
7 product banana, melon;
```

```
1 apple.weight
2 apple.price
3 banana.weight
4 banana.price
5 melon.weight
6 melon.price
```

# Exemplo – Aluno

```
❑ #include <stdio.h>
❑ #include <conio.h>
❑ struct estruturaAluno {
❑ char nome[20];
❑ int matricula;
❑ float nota;
❑ };
❑ int main() {
❑ struct estruturaAluno aluno;
❑ printf("Digite o nome:\n");
❑ scanf("%s", aluno.nome);
❑ printf("Digite a matricula:\n");
❑ scanf("%d", &aluno.matricula);
❑ printf("Digite a nota:\n");
❑ scanf("%f", &aluno.nota);
❑ if (aluno.nota > 6)
❑ printf("O aluno %s (%d) foi
aprovado.\n", aluno.nome,
aluno.matricula);
❑ else
❑ printf("O aluno %s (%d) foi
reprovado.\n", aluno.nome,
aluno.matricula);
❑ getch();
❑ return 0;
❑ }
```

# Vetores/Matrizes de registro

- Um registro (estrutura) é qualquer tipo de dado em C. Por isso, é possível criar vetores/matrizes do tipo criado pela estrutura.
- – Exemplo: `struct estruturaAluno alunos[10];`

# Função e Registro

- É possível passar só um elemento da estrutura ou toda a estrutura para a função.
- No primeiro caso, a utilização é igual a uma variável comum. No segundo caso, deve-se declarar como parâmetro da função apenas o nome da variável da estrutura a ser enviada.
- Exemplo: `void preencherNome (struct tipoAluno aluno);`



# Exemplo: Função e estrutura

```
□ #include <stdio.h>
□ #include <conio.h>
□ struct estruturaAluno {
□ char nome[20];
□ float nota;
□ };

□ void verifica (struct
 estruturaAluno a) {
□ if (a.nota>=6)
□ printf("O aluno %s foi
 aprovado!", a.nome);
□ else
□ printf("O aluno %s foi
 reprovado!", a.nome);
□ }

□ int main() {
□ struct estruturaAluno aluno;
□ printf("Digite o nome:\n");
□ scanf("%s", aluno.nome);
□ printf("Digite a nota:\n");
□ scanf("%f", &aluno.nota);
□ verifica(aluno);
□ getch();
□ return 0; }
```

# TypeDef

- É possível também renomear o nome dos tipos pré-definidos e também das estruturas criadas.
- Exemplo:
  - ▣ `typedef int Inteiro;`
  - ▣ `typedef struct estruturaAluno tipoAluno;`

# Exemplo: Typedef

```
#include <stdio.h>
#include <conio.h>
```

```
typedef struct estruturaAluno {
 char nome[20];
 float nota;
} tipoAluno;
```

```
struct estruturaAluno2 {
 char nome[20];
 float nota;
};
```

```
int main() {
 tipoAluno aluno;
 typedef struct estruturaAluno2 aluno2;
```

```
 aluno2 x;
 x.nota = 9;
 printf("Digite o nome:\n");
 scanf("%s", aluno.nome);
 printf("Digite a nota:\n");
 scanf("%f", &aluno.nota);
 if (aluno.nota > 6)
 printf("O aluno %s foi aprovado.\n", aluno.nome);
 else
 printf("O aluno %s foi reprovado.\n", aluno.nome);
 getch();
 return 0;
}
```

# Exercício - Registro



- Defina um registro empregado para guardar os dados (nome, sobrenome, data de nascimento, RG, data de admissão, salário) de um empregado de sua empresa. Defina um vetor de empregados para armazenar todos os empregados de sua empresa.
- Suponha que sua empresa tenha 5 empregados, leia os dados de todos e depois imprima.